OULUN AMMATTIKORKEAKOULU

Dmitri Akhonen

DYNAMOMETRIN MODERNISOINTI

DYNAMOMETRIN MODERNISOINTI

Dmitri Akhonen
Opinnäytetyö
Kevät 2015
Automaatiotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Automaatiotekniikan koulutusohjelma

Tekijä: Dmitri Akhonen
Opinnäytetyön nimi: Dynamometrin modernisointi
Työn ohjaajat: Matti Leino ja Timo Heikkinen
Työn valmistumislukukausi ja -vuosi: Kevät 2015 Sivumäärä: 38

Työn toimeksiantajana toimi Protacon Oy, joka on osa sähkö-, automaatio- sekä tietotekniikan ratkaisuja tarjoavaa Protacon Groupia. Protaconilta työn oli tilannut Dinex Ecocat Oy, joka on katalysaattoreiden valmistukseen erikoistunut yritys.

Insinöörityön tarkoituksena oli toteuttaa testilaitteisto katalysaattoreiden testausta varten. Testilaitteistoon kuuluvat anturit, savukaasuanalysaattorit, moottori ja pyörrevirtajarru. Laitteistoa ohjataan National Instrumentsin valmistaman automaatio-ohjaimen ja PC:n avulla. Laitteiston tarkoituksena on ajaa ennalta ohjelmoituja syklejä, jotta päästöt voitaisiin mitata käyttäjän haluamissa tilanteissa ja testit olisivat tarkasti toistettavissa.

Tehtävä toteutettiin käyttäen CompactRIO-sarjan automaatio-ohjainta ja sen ohjelmointiin tarkoitettua LabVIEW-kehitysympäristöä. Työssä käydään läpi hyödynnetyt tiedonsiirtomenetelmät, kuvaillaan kehitysympäristöä ja tutustutaan valmiin ohjelman käyttöliittymään. Työssä käsitellään myös laitteiston käyttöönottoa ja siinä ilmenneitä ongelmia.

Asiasanat: LabVIEW, dynamometri, automaatio

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Automation Engineering

Author: Dmitri Akhonen
Title of thesis: Modernisation of Dynamometer
Supervisors: Matti Leino and Timo Heikkinen
Term and year when the thesis was submitted: Spring 2015 Pages: 38

This thesis was done for Protacon Oy, which is a part of a Finnish technology engineering and service company Protacon Group. Equipment was ordered from Protacon by Dinex Ecocat Oy, a company that specializes in designing and developing of exhaust emission control systems.

The aim of this thesis was to design an equipment for testing of catalytic converters. Test system consists of a diesel engine and an eddy current brake. System also includes sensors and gas analyzers. The system is controlled by a PC and programmable automation controller manufactured by National Instruments. The purpose of this system is to run preprogrammed test cycles so that emissions could be measured in user defined conditions and tests could be repeated precisely in the same way every time.

The task was accomplished using CompactRIO-series automation controller which was programmed in the LabVIEW development environment. This thesis explains used data transfer methods, introduces development environment and takes a look at user interface of the complete program. Also installation of the system and encountered problems are described.

Keywords: LabVIEW, dynamometer, automation

ALKULAUSE

Työn toimeksiantajana toimi Protacon Oy, ja automaatiolaitteisto toimitettiin Dinex Ecocat Oy:n Vihtavuoren tehtaalle. Opinnäytetyön valvojana toimeksiantajan puolelta toimi Matti Leino ja ohjaajana lehtori Timo Heikkinen, joita haluisin kiittää tämän työn onnistumisesta. Kiitän myös lehtori Tuula Hopeavuorta työn tarkistamisesta.

Kiitoksen ansaitsevat myös Protaconin Miika Vehmanen, Erkki Toropainen ja konsulttina toiminut Timo Ukura, joiden arvokkaat neuvot auttoivat ohjelmointiympäristön oppimisessa ja työn valmistumisessa. Lisäksi kiitos kuuluu Dinex Ecocatin laboratorion henkilökunnan Arto Viitaselle ja Olli-Pekka Hyppöselle.

Oulussa 1.6.2015

Dmitri Akhonen

SISÄLLYS

LYHENTEET

AO	Analog Output, automaatio-ohjaimen analoginen lähtö
ASCII	American Standard Code for Information Interchange, merkistö
CAN	Control Area Network, automaatioväylä
cRIO	CompactRIO, automaatio-ohjain
DI	Digital Input, automaatio-ohjaimen digitaalitulo
DO	Digital Output, automaatio-ohjaimen digitaalilähtö
ECU	Electronic Control Unit, sähköinen ohjausyksikkö
FPGA	Field-programmable gate array, ohjelmoitava mikropiiri
I/O	Input/Output, tulo/lähtö
NI	National Instruments, automaatiolaitteita valmistava yritys
PAC	Programmable Automation Controller, ohjelmoitava automaatio-ohjain

1 JOHDANTO

Työn tilaajana toimi Dinex Ecocat Oy, joka halusi uudistaa yhden katalysaattoreiden testauspaikoistaan aiempaa monipuolisemmaksi. Tämä vaati uuden ohjausjärjestelmän ja käyttöliittymän suunnittelua.

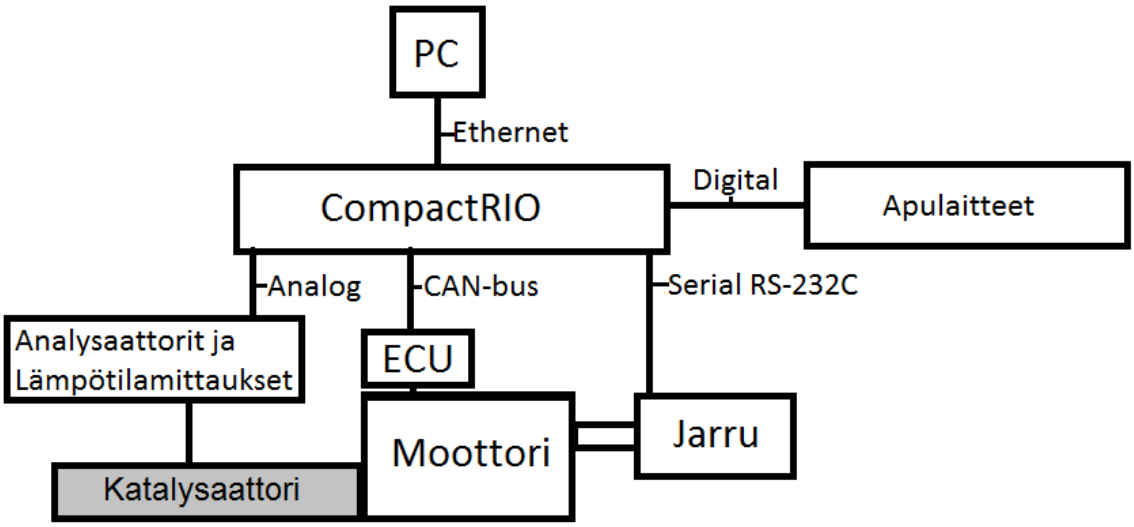
Ecocat on erikoistunut metallirakenteisten katalysaattoreiden valmistukseen sekä metallisten ja keraamisten partikkelifilttereiden pinnoituksiin. Ecocat siirtyi kesäkuussa 2013 tanskalaisen Dinexin omistukseen. Dinex on kansainvälinen yritys, joka kehittää ja valmistaa integroituja pakokaasukäsittelyn ratkaisuja raskaan kaluston tarpeisiin. (1, s. 2.)

Testauslaitteistolta vaadittiin kykyä ohjata dieselmoottoria ja jarrua haluttujen testausolosuhteiden saavuttamiseksi pakokaasumittauksia varten. Tämän lisäksi järjestelmän piti pystyä tallettamaan moottorilta ja pakokaasuanalysaattoreilta tulevia mittauksia. Ajettavat testisyklit ohjelmoidaan etukäteen, mutta järjestelmää piti pystyä ohjaamaan myös käsin.

Opinnäytetyön tavoitteena oli toteuttaa vaatimuksia vastaava ohjelma ja käyttöliittymä LabView-kehitysympäristön ja National Instrumentsin CompactRIO-sarjaan pohjautuvan laitteiston avulla.

2 TESTIPAIKAN LAITTEISTO

Testipaikan kokoonpanoon kuuluvat PC, ohjelmoitava automaatio-ohjain (PAC), savukaasuanalysaattorit, moottori, jarru, apulaitteita ja antureita. Testauspaikan periaatekaavio on esitetty kuvassa 1.



KUVA 1. Testauspaikan periaatekaavio

Käyttöliittymä ja mittausten tallennus toimivat PC:llä. Käyttöliittymä koostuu kahdesta ohjelmasta. Cycle Manager -ohjelmalla luodaan testisyklejä ja sitä voidaan käyttää tarvittaessa muuallakin kuin testipaikan tietokoneella, minkä jälkeen luodut syklit siirretään testipaikalle. Moottoripaikka2-ohjelma taas sisältää käyttöliittymän testauslaitteiston hallintaa ja valvontaa varten ja se toimii vain testipaikan tietokoneella, joka on yhdistetty ohjattavaan laitteistoon.

Laitteiden ohjausautomaatio toimii National Instrumentsin valmistamalla CompactRIO-automaatio-ohjaimella, joka kommunikoi ohjattavien laitteiden kanssa käyttäen CAN-väylää, sarjaliikennettä sekä perinteisiä automaation standardiviestejä, kuten 4–20 mA:n virtaviesti ja 0–10 V:n jänniteviesti.

Jarrun tehtävänä on vastustaa moottorin akselin pyörimistä, jotta haluttu kierrosnopeus tai momentti saadaan pidettyä halutussa arvossa. Näin moottorin päästöistä saadaan tietoa erilaisissa tilanteissa ja katalysaattoreita pystytään vertaamaan keskenään, koska testisykli on toistettavissa samanlaisena.

2.1 CompactRIO

Testilaitteiston ohjaamiseen valittiin NI cRIO-9066 -automaatio-ohjain, joka soveltuu tähän tarkoitukseen hyvin, koska sen ohjelmointimahdollisuudet ovat erittäin monipuoliset verrattuna ohjelmoitavaan logiikkaan ja ohjaimeen oli saatavilla kaikki tarvittavat laajennusmoduulit. Etuna on myös se, että käyttöliittymä ja ohjelma pystytään tekemään samassa kehitysympäristössä, mikä tekee niiden liittämisestä yhteen suhteellisen helppoa. Samantyyppisiä testilaitteistoja on yrityksessä toteutettu ennenkin National Instrumentsin ohjaimilla, joten vanhoista ohjelmista pystyi ottamaan mallia ja tietyiltä osin hyödyntää olemassa olevia ohjelman palasia, mikä myöskin helpotti valintaa.

NI cRIO-9066 sisältää kahdeksan paikkaa laajennusmoduuleita varten, joista kuusi otettiin käyttöön lopullisessa kokoonpanossa, jolloin automaatio-ohjaimeen jäi kaksi paikkaa laajennusvaraa. Laitteiston osien valinnassa auttoi CompactRIO Advisor -työkalu, jonka avulla eri ohjaimia sekä laajennusmoduuleita voi vertailla keskenään ja valita tarpeisiin parhaiten soveltuvat. Työkalun avulla voidaan valita myös liittimet, asennustarvikkeet, ohjelmistot ja palvelut. Kokoonpano on esitetty kuvassa 2.

cRIO-9066	Gigabit Ethernet
	Serial Port
	USB
NI 9862	CAN Interface, High-Speed
NI 9375	16-ch DI, 16-ch DO
NI 9217	PT100 RTD, 4-Ch
NI 9201	AI, +/-10 V, 8-Ch
NI 9203	AI, +/-20 mA, 8-Ch
NI 9263	AO, +/-10 V, 4-Ch

KUVA 2. CompactRIO-kokoonpano

CRIO-9066-ohjaimessa on 667 MHz:n tuplaydinprosessori, 256 megatavua keskusmuistia ja FPGA-piiri (Field Programmable Gate Array). Ulkoisten laitteiden kanssa voidaan kommunikoida USB-, sarja- ja Ethernet-portteja käyttäen.

2.2 Jarru ja moottori

Testilaitteiston moottorina toimii kuusisylinterinen Sisu 84 CTA -dieselmoottori, joka on valmistajan mukaan suunniteltu vaativiin työkonesovelluksiin. Moottoria ohjaa SisuTronic EEM3 -mallinen moottorinohjausyksikkö. (2, s. 1.) Moottorinohjausyksiköstä käytetään joskus lyhennettä ECU (Engine Control Unit), vaikka yleisesti ECU tarkoittaa sähköistä ohjausyksikköä (Electronic Control Unit), jonka tarkoitus voi olla muu kuin moottorin ohjaaminen. Moottorinohjausyksiköstä käytetään myös lyhenteitä ECM (Engine Control Module) ja EMS (Engine Management System). Tässä työssä moottorinohjausyksikköön viitataan kuitenkin lyhenteellä ECU. (3.)

ECU:n kanssa kommunikoidaan CAN-väylää käyttäen, mikä mahdollistaa moottorin mittausten tuomisen käyttöliittymään. ECU lähettää moottorin toiminnan kannalta tärkeitä tietoja kuten lämpötilan, paineen sekä moottorin pyörimisnopeuden ja vääntömomentin mittausarvoja. Moottorinohjausyksikön kautta pystytään antamaan myös ohjauskäskyjä moottorille.

Tässä työssä moottorin ohjaukset rajoittuvat lähinnä nopeusohjeeseen, joka annetaan kierroksina minuutissa (rpm). ECU:n oma säätöpiiri huolehtii nopeusohjeessa pysymisestä. Alun perin oli tarkoitus toteuttaa vaihtoehtoinen moottorin ohjaustapa, joka olisi toiminut siten, että moottorinohjausyksikölle olisi annettu kaasupolkimen asento analogiaviestillä ja syklien luontiohjelmassa olisi saanut valita, kumpaa ohjaustapaa halutaan käyttää. Tilaajan tutkittua moottoria tarkemmin selvisi, että ECU ei analogiaohjaukseen reagoinut, joten se jätettiin kokonaan pois. Kuvassa 3 on esitetty moottorin tekniset tiedot.

ENGINE TYPE	66 CTA	74 CTA	84 CTA
Rated power (kw)	98 - 175	135 - 215	187 - 305
Nominal speed (rpm)	2200	2200	2200
Rated torque (Nm)	550 - 1000	670 - 1150	1185 - 1600
At speed (rpm)	1500	1500	1500
Number of cylinders	6	6	6
Displacement (Litres)	6,6	7,4	8,4
Cylinder bore (mm)	108	108	111
Stroke (mm)	120	134	145
Rotation direction (seen from flywheel end)	CCW	CCW	CCW
Aspiration	Turbocharged and charged air cooled		
Type of intercooler	Air to air	Air to air	Air to air
Emission certification	EU Stage 3 A / EPA Tier3		
Injection system	Common Rail	Common Rail	Common Rail

KUVA 3. Sisu 84 CTA -moottorin tekniset tiedot (2, s. 2)

Testipaikan jarrua ohjataan Froude Hoffmanin valmistaman Texcel V4-EC -järjestelmän avulla. EC mallinimen lopussa viittaa siihen, että jarru on pyörrevirtatoiminen eli jarruttaminen tapahtuu sähköisesti. Jarruista on olemassa myös hydrauliikalla toimivia versioita. Lisäksi on olemassa malleja, jotka sisältävät liitännät ja säätimet myös moottorin ohjausta varten. Tämä malli ei kuitenkaan ollut sellainen, mikä lisäsi ohjelmointityötä. Järjestelmän kanssa kommunikoidaan käyttäen RS-232C-standardiin pohjautuvaa sarjaporttia ja ASCII-koodattuja viestejä. Yhteyden avulla jarrulle pystytään välittämään vääntömomentin asetusarvo ja lukemaan nopeus- ja vääntömittaukset.

3 LAITTEIDEN VÄLINEN KOMMUNIKOINTI

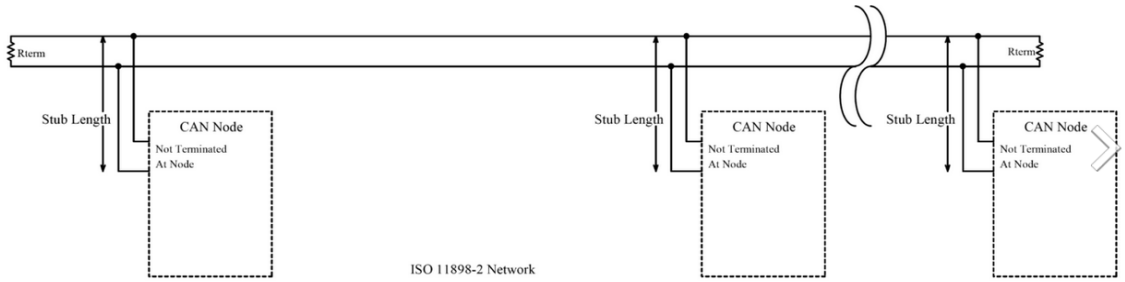
Tässä työssä käytettiin erilaisia kommunikointimenetelmiä, joista eniten perehtymistä vaati sarjaliikenne ja CAN-väylä. Sarjamuotoista liikennettä käyttävät useat väylät, mutta tässä työssä sarjaliikenteellä tarkoitetaan fyysistä RS-232C-liityntää ja ASCII-koodausta käyttävää liikennettä. Kuten edellä on sanottu, CAN-väylää käytettiin kommunikointiin moottorinohjausyksikön kanssa ja sarjaliikennettä yhteyden luomiseen jarrun ohjausjärjestelmään.

3.1 CAN-väylä

CAN-väylä on alun perin Boschin vuonna 1985 kehittämä liityntätekniikka autoteollisuuden tarpeisiin, jolla haluttiin vähentää johdotuksia auton elektronisten laitteiden välillä. Tekniikka alkoi yleistyä ja vuonna 1993 siitä tuli kansainvälinen standardi ISO 11898. Myöhemmin standardoitiin useita CAN-tekniikkaan pohjautuvia ylemmän tason protokollia kuten CANopen ja DeviceNet. Tämän jälkeen protokollat otettiin käyttöön myös muilla teollisuudenaloilla. Seuraava teksti keskittyy lähinnä autosovelluksiin, sillä siihen CAN-väylää käytettiin tässä työssä. (4.)

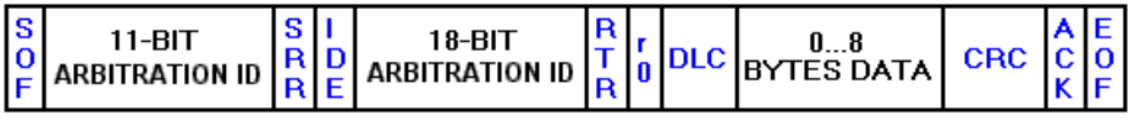
Yksi CAN-väylän eduista on se, että johdotukset moottorinohjausyksikön ja laitteiden välillä vähenevät, koska analogia- ja digitaaliviestejä ei tarvitse viedä jokaista omilla johdoillaan. CAN-verkossa kaikki laitteet näkevät kaikki viestit, joista poimitaan laitteen kannalta tarpeelliset. Tämä helpottaa verkon muokkaamista ja diagnostiikkaa, sillä kuuntelevan laitteen lisääminen ei häiritse verkon toimintaa. Verkossa on käytössä priorisointimenetelmä, jotta viestit eivät menisi päällekkäin, jos kaksi tai useampi laite lähettää viestiä yhtä aikaa. Menetelmä toimii siten, että alemman prioriteetin laite keskeyttää lähetyksensä ja yrittää myöhemmin uudestaan huomatessaan, että korkeamman prioriteetin laite lähettää viestiä. Virheentarkistusta varten CAN käyttää CRC:tä (Cyclic Redundancy Code), jonka avulla tarkistetaan jokaisen kehyksen sisällön oikeellisuus. (4.)

CAN-väylän yhteydessä voidaan käyttää useita erilaisia fyysisiä kerroksia, joista valitaan tarkoitukseen parhaiten soveltuva. Kerrosten erona voi olla esimerkiksi nopeus, kytkentä ja vikasietokyky. High-Speed CAN (CAN C tai ISO 11898-2) on näistä kerroksista käytetyin ja sitä käytettiin tässäkin työssä. High-Speed CAN -verkossa käytetään kahta johdinta ja päätevastuksia väylän päissä, kuten kuvasta 4 käy ilmi. Usein väylään tarvitaan vielä syöttöjännite CAN-laitteiden lähetin-vastaanotinpiireille, koska ne voivat olla optisesti erotettuja laitteen muusta elektroniikasta. Väylään syötettävä jännite on tyypillisesti 7–30 V. High-Speed CAN -verkon tiedonsiirtonopeus on maksimissaan 1 Mbit/s, mutta nopeus laskee, mikäli käytetään pitkiä kaapeleita. (4.; 5.)



KUVA 4. High-Speed CAN -verkon kytkentä (5)

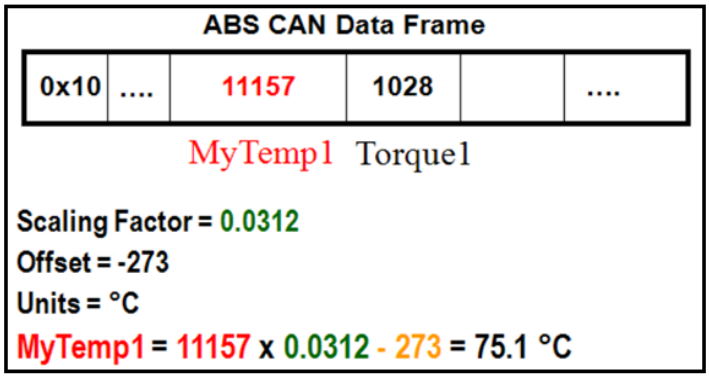
CAN-laitteiden lähettämää datapakettia kutsutaan kehykseksi. Kehyksistä on käytössä kahta eri formaattia, standard ja extended. Näiden formaattien erona on Arbitration ID:n pituus, joka standardissa on 11 bittiä ja extendedissä 29 bittiä. Kuvassa 5 on esitetty extended-formaatin kehys.



KUVA 5. Extended-formaatin kehys (4)

Kehittäjän kannalta tärkeimmät kentät ovat Data ja Arbitration ID. Data-osa nimensä mukaisesti sisältää itse lähetettävän datan, joka on muodostettu valmistajan spesifikaation mukaan, joka useimmiten ei ole vapaasti saatavilla.

Yksinkertainen esimerkki datan purkamisesta on esitetty kuvassa 6.



KUVA 6. Raakadatan purkaminen valmistajan antamien parametrien avulla (4)

Arbitration ID:llä taas on kaksi tehtävää. ID kertoo laitteille, joita CAN-termein kutsutaan nodeiksi, mikä viesti on kyseessä. ID on kaikilla viesteillä yksilöllinen. Samalla ID määrittelee viestin prioriteetin, koska CAN-väylässä pienimmällä ID-arvolla on suurin prioriteetti. Teknisestä näkökulmasta priorisointi on toteutettu siten, että verkon nodet lukevat kaikki bitit yhtä aikaa, minkä lisäksi on määritelty, että looginen 0 on hallitseva bitti ja 1 alistuva. Tämän menetelmän avulla hallitsevan bitin lähettävä node saa etuajo-oikeuden, minkä jälkeen alistuvan bitin lähettäjä keskeyttää lähetyksensä ja yrittää myöhemmin uudestaan. Toimintaa on havainnollistettu kuvassa 7. Muiden kehyksen kenttien tarkoitus on selitetty lyhyesti taulukossa 1. (5, s. 1.)

	Start Bit	ID Bits										The Rest of the Frame
		10	9	8	7	6	5	4	3	2	1	0
Node 15	0	0	0	0	0	0	0	0	1	1	1	1
Node 16	0	0	0	0	0	0	0	1	Stopped Transmitting			
CAN Data	0	0	0	0	0	0	0	0	1	1	1	1

KUVA 7. CAN-väylän priorisointimenetelmä (5, s. 1)

TAULUKKO 1. CAN-kehyksen kentät (4)

SOF	**Start of frame bit**	**Aloitusbitti, looginen 0**
IDE	**Identifier extension bit**	**Kehyksen formaatti, looginen 0, jos 11-bit**
RTR	**Remote transmission request bit**	**Kertoo, onko kyseessä lähetyspyyntö- vai dataviesti**
DLC	**Data length code**	**Datakentän tavujen määrä**
CRC	**Cyclic redundancy check**	**Tarkistuskoodi virheentunnistusta varten**
ACK	**Acknowledgment slot**	**Kuittausbitti**

3.2 ASCII-muotoinen sarjaliikenne

ASCII-muotoinen sarjaliikenne koostuu fyysisestä määrittelystä, liikennöintiasetuksista ja ASCII-koodauksesta. Tässä työssä käytettiin RS-232C-määrittelyä, joka käsittää liitännässä käytettävät jännitetasot ja pinnien järjestyksen. Standardi ei kuitenkaan ota kantaa liikenteessä käytettävään symboleiden koodaustapaan. Koodaustapana käytetään kuitenkin yleensä ASCII-merkistöä, jossa jokaiselle merkille on määritelty tietty bittikuvio, joka muunnetaan RS-232-tasoiseksi jännitesignaaliksi. Sarjaliikenteessä on myös tärkeää muistaa valita oikeat liikennöintiasetukset, jotta viestit siirtyisivät laitteiden välillä.

3.2.1 RS-232

RS-232 on vuonna 1962 esitelty standardi, joka on alun perin suunniteltu kommunikointiin kaukokirjoituslaitteiden ja modeemien välillä. Tämän jälkeen standardiin on tullut useita lisäyksiä yhteensopivuuden varmistamiseksi. Standardia on yleisesti käytetty tietokoneiden sarjaportissa, johon pystytään liittämään erilaisia oheislaitteita. Moderneissa kotitietokoneissa USB-liitäntä on syrjäyttänyt sarjaportin, koska USB-liitännän liitin on pienempi ja nopeus suurempi kuin sarjaportilla. RS-232-standardi sallii kuitenkin suuremmat kaapeleiden pituudet ja kommunikointi on yksinkertaisempaa kuin USB:n avulla.

Tämän vuoksi RS-standardi on säilyttänyt asemansa teollisuus- ja tieteislaitteissa. Vaativammissa sovelluksissa käytetään muita standardeja kuten RS-485. (6.)

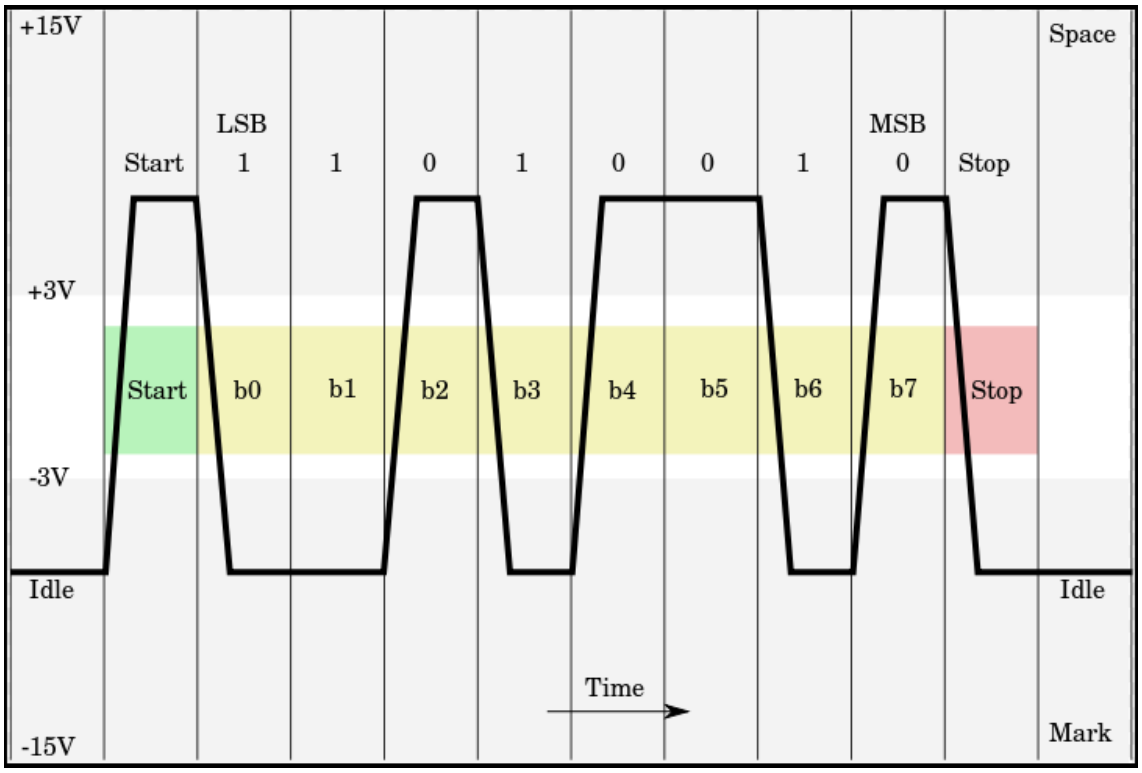
RS-232-yhteensopiva sarjaportti käyttää nimensä mukaisesti sarjamuotoista liikennettä kommunikointiin. Tämä tarkoittaa, että tiedonsiirto tapahtuu yksi bitti kerrallaan peräkkäin. Tiedon lähetykselle (TxD) ja vastaanotolle (RxD) on omat johtimet, joten liikenne toimii molempiin suuntiin samanaikaisesti eli niin kutsutussa full-duplex-tilassa. Sarjaporttilaitteet on jaettu DTE- ja DCE-kategorioihin. Lyhenteet tulevat sanoista data terminal equipment ja data circuit-terminating equipment. Käytännön tasolla tämä tarkoittaa, että kytkennöissä kannattaa olla erityisen tarkkana, jotta laitteet saadaan kytkettyä toimivasti yhteen. Jos laitepari on tyyppiä DTE-DCE, kytkentä onnistuu tavallisesti käyttämällä suoraa kaapelia. Usein voi kuitenkin olla tarve yhdistää kaksi DTE-laitetta, kuten esimerkiksi kaksi tietokonetta. Tällöin on käytettävä ristiin kytkettyä null modem -kaapelia. Varsinkin vanhojen sarjaporttia käyttävien laitteiden kanssa kannattaa olla erityisen tarkkana kytkennän suhteen, sillä joissaan laitteissa voi esiintyä epästandardeja kytkentöjä ja on olemassa muitakin kuin edellä esitettyjä kaapeleita. Kyseessä voi olla myös kokonaan eri standardia ja jännitettä käyttävä portti, vaikka se päällepäin näyttäisi RS-232-portilta. Tästä johtuen on aina turvauduttava valmistajan manuaaleihin. (7.)

Alun perin standardi määritteli fyysisen liittimen 25-pinniseksi ja liitin sisälsi 20 erilaista signaalikytkentää. Valmistajat siirtyivät kuitenkin käyttämään 9-pinnistä DE-9-liitintä kustannus- ja tilasyistä. Yksinkertaisimmillaan kaksisuuntainen liikenne toimii kolmella johtimella: RxD, TxD ja Ground. Muut portin pinnit ovat varattuja vuonohjaukselle, jonka avulla liikennettä voidaan hidastaa, esimerkiksi vastaanottavan laitteen lukupuskurin ollessa täyttymässä. Laitepohjaista vuonohjausta ei ole pakko käyttää, sillä vuonohjaus voidaan toteuttaa tarvittaessa myös ohjelmallisesti, tai jättää kokonaan pois. (6.)

3.2.2 Liikennöintiasetukset

Sarjaporttia käytettäessä täytyy määritellä myös muutamia asetuksia, jotta liikenne toimisi. Näitä ovat nopeus, databitit, pariteetti ja pysäytysbitit. Sarjaporttien nopeus ilmoitetaan baudeina eli symboleina sekunnissa. Nopeuden on oltava asetettu molemmissa laitteissa samaksi. Data bits ilmaisee, montako bittiä on varattu yhdelle kirjaimelle, ja asetus voi olla jotain 5:n ja 9:n väliltä. Useimmiten kuitenkin käytetään kahdeksaa bittiä. Pariteetti taas on yksinkertainen virheentarkistustapa, jonka ideana on lisätä databittien perään 1- tai 0-bitti siten, että yhteenlaskettujen 1-bittien määrä on pariton (odd) tai parillinen (even) riippuen asetuksesta. Tämän avulla vastaanottava laite havaitsee virheen, koska bittien parillisuus on virhetilanteessa eri kuin vastaanottava laite olettaa. Menetelmän avulla virheitä ei voida korjata vaan data on lähetettävä uudestaan, koska ei voida erottaa, mikä viestin bitti on muuttunut. Menetelmä ei myöskään ole erityisen luotettava, koska parillisen määrän virheitä sattuessa virhettä ei havaita. Usein pariteettia ei oteta käyttöön ja virheentunnistus hoidetaan tietoliikenneprotokollan avulla. Pysäytysbitin tehtävänä on kertoa vastaanottavalle laitteelle lähetyksen loppumisesta. Nykyisin käytetään enimmäkseen yhtä pysäytysbittiä. Asetukset voivat olla merkitty esimerkiksi muodossa 8N1 eli kahdeksan databittiä, ei pariteettia ja yksi pysäytysbitti. (7.; 8.)

Laitteiden välinen kommunikointi tapahtuu useimmiten käyttämällä ASCII-muotoisia merkkijonoja, jotka muunnetaan siirtoa varten RS-232-tasoiseksi jännitesignaaliksi. Kuvassa 7 on esitetty kuvitteellinen mittaus ASCII-enkoodatusta K-kirjaimesta.



KUVA 7. ASCII-muotoinen K-kirjain (0x4B) jännitteenä (6)

4 LABVIEW-KEHITYSYMPÄRISTÖ

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) on National Instrumentsin tuottama kehitysympäristö, jonka avulla pystytään suunnittelemaan sekä toteuttamaan erilaisia mittaus-, automaatio- ja testaussovelluksia. Kehitysympäristössä käytetään graafista ohjelmointikieltä nimeltä G. G-kieli on niin kutsuttu dataflow programming -kieli, jonka suoritus tapahtuu graafisen lohkokaavion pohjalta. Kaavio sisältää erilaisia yhteen johdotettuja komponentteja, jotka sisältävät tiettyjä funktioita. Komponentin funktiot suoritetaan heti, kun kaikki tarvittava sisäänmenodata on saatavilla. G-kieli osaa hyödyntää tehokkaasti rinnakkaisuutta ja ajaa useita koodikokonaisuuksia rinnakkain. Tästä yksinkertaisena esimerkkinä voisi olla ohjelma, jossa yksi while-silmukka hoitaa mittausten lukemisen ja kirjoittamisen lokiin, ja toinen vastaa käyttöliittymästä. Näin tiedonkeruu ei pysähdy, vaikka jälkimmäinen silmukka jäisi odottamaan käyttäjän syötettä. Silmukat voivat kuitenkin kommunikoida keskenään ja tietoa keräävän silmukan toimintaa voidaan muuttaa syötteen perusteella. (9.)

Rinnakkaiset prosessit vaativat kehittäjältä erityistä tarkkuutta varsinkin kilpailutilanteiden huomioimisessa (race condition). Kilpailutilanne on sellainen, että kaksi rinnakkaista prosessia suoritetaan eri järjestyksessä kuin on oletettu. Tämä tilanne on esitetty kuvassa 8. Vasemmanpuoleisessa taulukossa on esitetty toiminta, jota ohjelmalta odotetaan. Oikealla puolella taas on esitetty tilanne, jossa säikeiden (Thread) suoritusjärjestys on muuttunut puuttuvan ajoituksen tai lukituksen takia. Toinen säie on tällä suorituskierroksella alkanut kasvattaa yhteistä muuttujaa, ennen kuin ensimmäinen on kasvattanut sitä, mikä aiheuttaa virheen ohjelmassa. Kilpailutilanteet voi monesti olla vaikea löytää, koska lopputulos saattaa vaihdella suorituskertojen välillä, ja tilanne on vaikeasti toistettavissa. Tämän vuoksi ohjelmassa täytyy varmistaa, että suoritusjärjestys pysyy jokaisella kerralla oikeana. (10.)

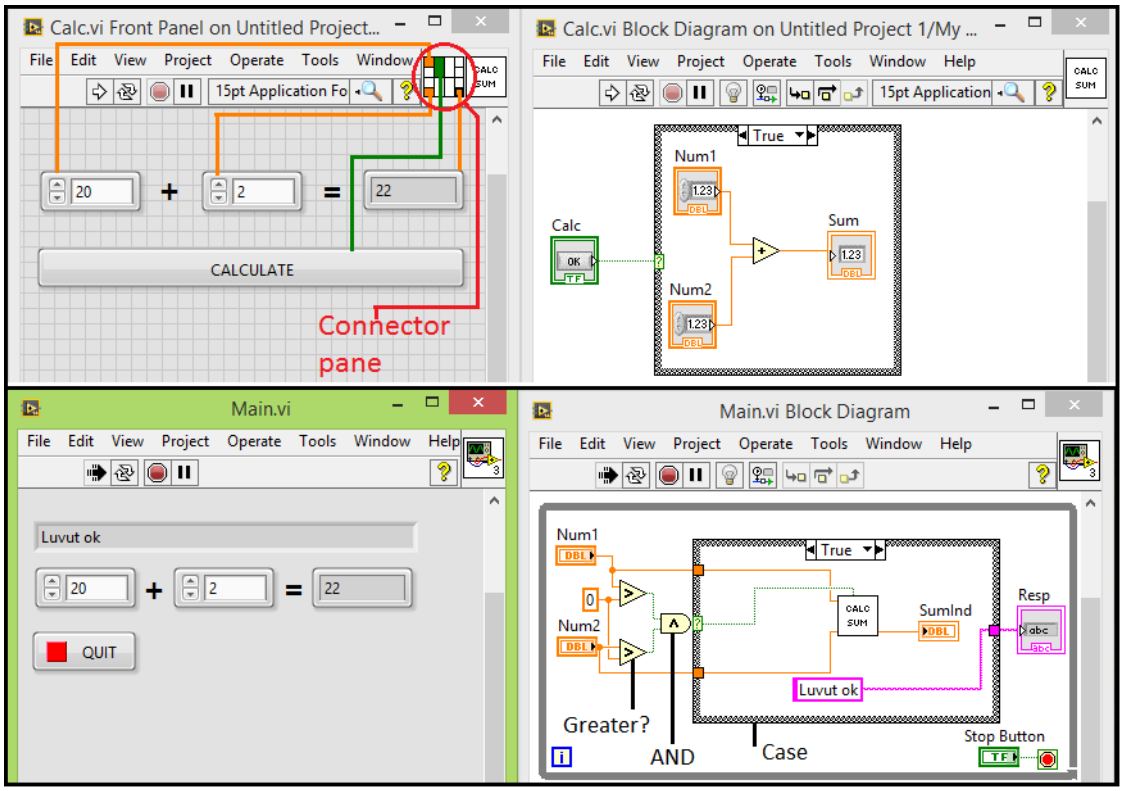
Thread 1	Thread 2		Integer value
			0
read value		←	0
increase value			0
write back		→	1
	read value	←	1
	increase value		1
	write back	→	2

Thread 1	Thread 2		Integer value
			0
read value		←	0
	read value	←	0
increase value			0
	increase value		0
write back		→	1
	write back	→	1

KUVA 8. Kilpailutilanne kahden säikeen välillä (10)

LabVIEW ympäristössä ohjelman toiminnallisuuden tuottaminen ja käyttöliittymän luominen on sidottu toisiinsa. Tämä on toteutettu siten, että jokaisella ohjelmalla (VI) ja alirutiinilla (SubVI) on kolme osaa: front panel, block diagram ja connector pane. Front panel on käyttöliittymä, joka näkyy käyttäjälle. Käyttöliittymän objektit ovat ohjauksia (controls) ja ilmaisimia (indicators). Kun käyttöliittymään pudotetaan objekteja, kuten nappeja tai mittareita, ne ilmestyvät myös block diagrammiin, missä niille voidaan ohjelmoida haluttu toiminta. Connector panen avulla ohjelma voidaan kytkeä aliohjelmaksi osaksi isompaa kokonaisuutta. Siihen merkitään ohjaukset, jotka halutaan tuoda ohjelman sisään ja ilmaisimet, jotka halutaan viedä ulos. Tämä on kätevää kahdesta syystä. Käyttöliittymä syntyy ohjelmoinnin ohessa ja aliohjelmat pystytään helposti testaamaan käyttöliittymänäkymästä liittämättä niitä osaksi isompaa kokonaisuutta. (9.)

Asiaa on selvennetty vielä kuvassa 9, jossa ylempänä ohjelma toimii itsenäisesti ja alempana se on lisätty pääohjelmaan, missä sen ympärille on rakennettu lisää toiminnallisuutta.



KUVA 9. VI toiminnassa itsenäisesti ja osana kokonaisuutta

LabVIEW sisältää suuren määrän erilaisia kirjastoja tiedonkeruuta, laskentaa ja signaalikäsittelyä varten. Ympäristö sisältää myös laajan tuen erilaisille rajapinnoille, jotka mahdollistavat kaikenlaisten laitteiden ja instrumenttien ohjauksen. Tähän tarkoitukseen voidaan käyttää joko suoria väyläkomentoja tai joidenkin laitteiden kohdalla jopa valmiita LabVIEW-yhteensopivia ajureita. Erilaisille funktioille ja fyysisille moduuleille löytyy myös yleensä kattavat dokumentit ja esimerkkiohjelmat, jotka helpottavat ohjelmointia suosittelemalla hyviä menetelmiä. (9.)

5 OHJAUSSOVELLUS

5.1 Sovelluksen ohjelmointi

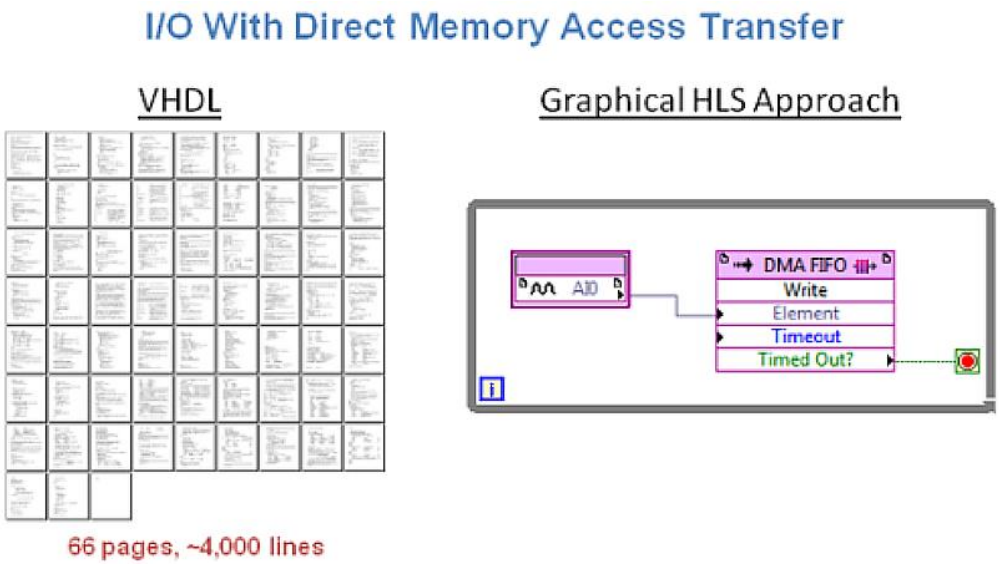
CompactRIO-laitteistoa odotellessa projekti aloitettiin tutkimalla aikaisemmin tehtyjä ohjelmia ja tekemällä pieniä kokeiluohjelmia, koska aiempaa ohjelmointikokemusta käytetystä ympäristöstä ei ollut. Kun laitteisto saapui, varsinainen ohjelmointi voitiin aloittaa.

Tässä tapauksessa CompactRIO-laitteistolla tehty sovellus on jaettu kolmelle tasolle: FPGA, Real-Time ja PC-pohjainen käyttöliittymä. Käyttöliittymää ajetaan PC:llä ja varsinainen automaatio cRIO-automaatio-ohjaimella, jossa osa toiminnallisuudesta on siirretty FPGA-piirille.

FPGA-piiri

FPGA eli Field Programmable Gate Array on piiri, joka sisältää rajallisen määrän loogisia portteja, jotka voidaan johdottaa uudelleen halutun toiminnan toteuttamiseksi. Loogisten porttien lisäksi piirissä on tulot ja lähdöt tiedonsiirtoa varten. FPGA on laitteistopohjaisen logiikan ansiosta erittäin nopea, sillä sovelluksen osien ei tarvitse kilpailla prosessorin suoritusajasta, eikä välissä ole ajureiden ja käyttöjärjestelmän viiveitä haittaamassa suoritusta. Näin sovelluksen vasteaika voi olla esimerkiksi niinkin pieni kuin 25 ns. (11.)

Haittapuolena on se, että ohjelman kääntäminen kestää kauan, mikä vaikeuttaa ohjelmointityötä, sillä pienimmänkin muutoksen testaaminen vaatii ohjelman uudelleen kääntämistä ja yli kymmenen minuutin odotusta. LabVIEW’lla ohjelmoitaessa aikaa säästetään kuitenkin siinä, että ei tarvitse mennä aivan matalalle ohjelmointitasolle ja vaativaakin toiminnallisuutta pystytään toteuttamaan suhteellisen yksinkertaisesti verrattuna esimerkiksi VHDL-suunnitteluun. Kuvassa 10 on verrattu kahta toiminnallisuutta toteutettuna VHDL- ja G-kielellä.



KUVA 10. Sama toiminta kuvattuna VHDL- ja G-kielellä (11)

Sovelluksen ohjelmointi aloitettiin FPGA-piiristä, sillä analogisten ja digitaalisten mittausten luku on nopeinta suorittaa siellä. Kokoonpanossa käytetyille moduuleille oli hyviä valmistajan tarjoamia esimerkkiohjelmia, joita pystyttiin hyödyntämään. Tässä vaiheessa kuitenkin tuli ensimmäinen ongelma. Kun kaikki mittaukset ja ohjaukset lisättiin FPGA-ohjelmaan, se ei kääntynyt. Lopulta ongelma onnistuttiin paikantamaan AO-moduuliin, joten muutokset ohjelman toimintaan saamiseksi keskitettiin sinne. Tämä ei kuitenkaan johtanut toivottuun lopputulokseen ja National Instrumentsiin otettiin yhteyttä. Ongelma kuvailtiin ja mukaan laitettiin yksinkertaistettu ohjelma, jossa ongelma ilmeni.

Asiakastuki löysi lopulta ongelman alkuperän. CompactRIO-automaatio-ohjaimessa tuloja voidaan lukea myös vaihtoehtoisella tavalla Scan Enginen avulla, jonka toimintaan ei perehdytä nyt tarkemmin. AO-moduulissa oli tähän liittyen vika, joka esiintyi, kun laitteistoa yritettiin käyttää hybriditilassa eli niin, että yksi moduuli käyttää Scan Engineä ja muut FPGA:ta. Tässä tapauksessa CAN-moduuli laukaisi kyseisen ongelman, sillä se käyttää toiminnassaan Scan Enginen komponentteja. Ongelma korjaantui asentamalla uusimmat ajurit sekä tietokoneeseen että automaatio-ohjaimeen. Uusien ajureiden olemassaolo jäi huomaamatta, sillä LabVIEW’n automaattinen päivitystyökalu ei niitä jostain syystä ehdottanut. Ongelmakuvaus löytyy nykyisin myös National Instrumentsin sivuilta lähdeviittauksessa mainitusta osoitteesta. (12.)

Kun FPGA-ohjelma lopulta kääntyi, kaikkien moduuleiden toiminta testattiin ja voitiin siirtyä itse sovelluksen tekoon. Tässä vaiheessa NI 9217-lämpötilamittausmoduuli aiheutti hieman hämmennystä, koska mittaustiedot näyttivät päivittyvän aivan liian hitaasti. Ongelma selvisi kuitenkin nopeasti ohjekirjan tarkemmalla lukemisella. Ohjekirjasta ilmeni, että moduulilla on kaksi toimintatilaa, joiden avulla voidaan painottaa tarkkuutta tai nopeutta. Tarkkuustilassa vasteaika oli kohtuullisen pitkä eli 200 ms/kanava. Tästä syystä päädyttiin nopeustilaan, jossa vasteaika putosi 2,5 millisekuntiin. Sovelluksella ei ollut niin kovat vaatimukset, että olisi ollut tarvetta hyödyntää FPGA:n tarjoamaa nopeutta, joten sinne ei tehty lisäyksiä vaan tyydyttiin mittaustietojen lukuun.

Sarjaliikenneajuri

Seuraavana aloitettiin Texcel-jarrun sarjaliikenneajurin ohjelmointi. LabVIEW’ssa on valmiita funktioita sarjaliikennekommunikointiin, joten aivan alimmalta tasolta liikennettä ei tarvinnut toteuttaa. Vastaavaa jarrua on käytetty ennenkin hieman yksinkertaisemmassa testipaikassa. Vanha ajuri otettiin pohjaksi ja lähdettiin kehittämään sitä eteenpäin. Tarvittavia lisäyksiä tässä vaiheessa olivat useamman toimintamoodin tuki ja parempi virheenhallinta esimerkiksi yhteyden katkeamisen varalta. Ajuri tarvitsi myös optimointia tietyissä tilanteissa, jotta nopeus säilytetään. RS-232-kappaleessa sivuttiin virheentarkistusta pariteetin yhteydessä. Tässä tapauksessa tehtiin niin, että pariteetti poistettiin käytöstä ja virheentunnistus hoidettiin ohjelmallisesti. Tämä toteutettiin valmistajan kuvaaman menetelmän mukaan, jossa viestin loppuun lisätään tarkistussummaksi ASCII-merkki, joka valitaan valmistajan antaman kaavan perusteella. Tämänkään menetelmän avulla virhettä ei voi korjata, vaan data täytyy hylätä ja odottaa uutta lähetystä. Menetelmä on kuitenkin luotettavampi kuin pariteetti, sillä virheet voidaan havaita, vaikka niiden määrä olisikin parillinen.

Jarru hoitaa tiedon lähetyksen siten, että haluttua tietoa voidaan kysyä tarvittaessa tai antaa käsky lähettää tietopakettia jatkuvasti. Tässä projektissa jatkuva lähetys oli tietysti kätevintä ainakin kierrosnopeus- ja vääntömomenttimittausten osalta. Ajuri toimii tilakone tyyppisesti eli se suorittaa ohjelmoitua sekvenssiä, josta voidaan kuitenkin tilanteen mukaan poiketa ja hypätä eri askeleeseen.

Näin voidaan toimia esimerkiksi kun huomataan, että yhteys jarruun on katkennut ja hypätään kirjoitusvaiheen sijasta alustusaskeleeseen, missä alkuasetukset yritetään syöttää uudestaan ja saada yhteys takaisin. Testausta varten CompartRIO liitettiin tietokoneeseen ristiinkytketyllä kaapelilla ja kirjoitettiin ohjelma, joka simuloi jarrun toimintaa. Tämän ansiosta joitain korjauksia pystyttiin tekemään ilman oikeaa jarrulaitteistoa.

Käyttöliittymä

Seuraavaksi siirryttiin tekemään käyttöliittymää. Tässä hyödynnettiin olemassa olevia ohjelman palasia käyttöliittymän ja Real-Time-sovelluksen luomisessa, siltä osin kuin se oli mahdollista. Tällä tavoin käyttöliittymän toimintaa voitiin jo esitellä, sillä tässä vaiheessa syklejä pystyttiin tekemään, lataamaan ja aloittamaan. Syklien luonti tapahtuu erillisellä Cycle Manageriksi nimetyllä ohjelmalla, joka ei ole suoraan sidoksissa testaussovellukseen ja sitä pystytään käyttämään tarvittaessa myös muilla tietokoneilla.

Kun käyttöliittymä oli esittelykunnossa, pidettiin asiakkaiden kanssa palaveri, jossa sovittiin tarvittavat muutokset ja lisäykset, joista kerrotaan tarkemmin Käyttöliittymä-kappaleessa. Palaverissa saatiin myös selvyyttä muutamiin asioihin, mikä helpotti työn jatkamista.

CAN-ajuri

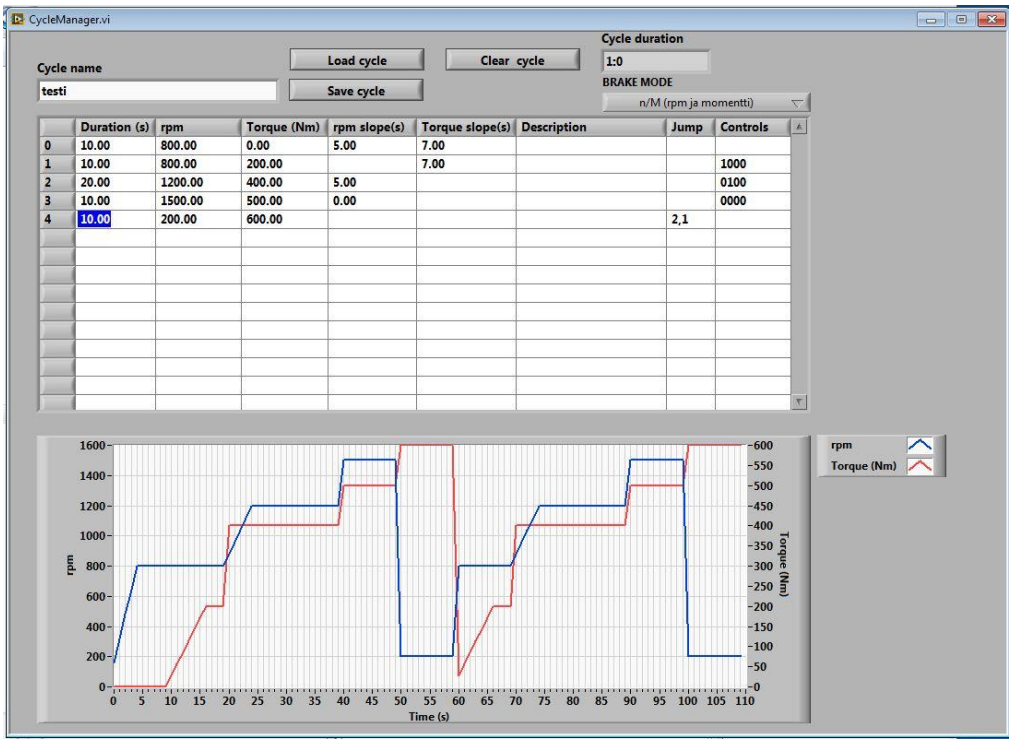
Seuraavana vuorossa oli CAN-ajurin ohjelmointi. Myös siihen oli olemassa valmiita ohjelman palasia, joita pystyi soveltamaan. Eräässä projektissa oli käytetty hieman uudempaa Sisun moottorinohjausyksikköä, joten viestien purkaminen selkokielisiksi oli pääosin valmiina. Muutoksia kuitenkin jouduttiin tekemään, sillä tällä kertaa käytössä ollutta NI 9862 CAN-moduulia ohjelmoidaan XNET-rajapinnalla, joka on eri kuin vanhemmassa projektissa käytetty. XNET-rajapinnassa CAN-kehys on hieman erilainen, minkä vuoksi tarvittiin muutoksia tiedon lukuun, jotta viestit saatiin purkuohjelman haluamaan muotoon. Ajurin tekeminen vaati tutustumista uuteen rajapintaan myös muilta osin, sillä yhteyden luonti toimii eri tavalla kuin edellisessä ohjelmassa. CAN-yhteyttä testattiin vastaavalla tavalla kuin sarjaliikennettä. Avuksi otettiin USB-liitännällä varustettu

CAN-portti, joka liitettiin tietokoneeseen, minkä jälkeen tehtiin yksinkertainen ohjelma, jonka avulla kirjoitettiin ja luettiin CAN-viestejä. Kuten aiemmin mainittu, CAN-yhteyden avulla vastaanotetaan ECU:n lähettämiä mittaustietoja moottorin toiminnasta ja lähetetään moottorille kierrosohje. Sovellukseen ei tarvinnut lisätä PID-säätimiä, sillä jarrun ja moottorin omat säätimet pystyvät pitämään moottorin vääntömomentin ja kierrokset annetuissa ohjearvoissa.

Kun ohjelman ydintoiminnot ja kommunikointi olivat pitkällä, siirryttiin kehittämään käyttöliittymää eteenpäin, testaamaan ja viilaamaan ydintoimintojen toimintaa paremmaksi. Tämä vaihe sisälsi myös raportoinnin lisäämisen. Jokaisen ajon mittaustiedot tallennetaan erilliseen tiedostoon, jota voidaan jälkeenpäin tarkastella taulukkolaskentaohjelmalla. Mittaustietojen tallennukseen olisi voinut käyttää myös tietokantaa, mutta asiakkaalle sopi edellä mainittu menetelmä paremmin.

5.2 Käyttöliittymä

Ohjelmien käyttöliittymä pyrittiin pitämään mahdollisimman yksinkertaisena ja asiakkaan toiveet otettiin huomioon. Tässä kappaleessa esitellään käyttöliittymä ja sen toiminnot. Esittely aloitetaan Cycle Manager -ohjelmasta, jonka avulla halutut ajosyklit luodaan. Ohjelman päänäkymä on esitetty kuvassa 11.

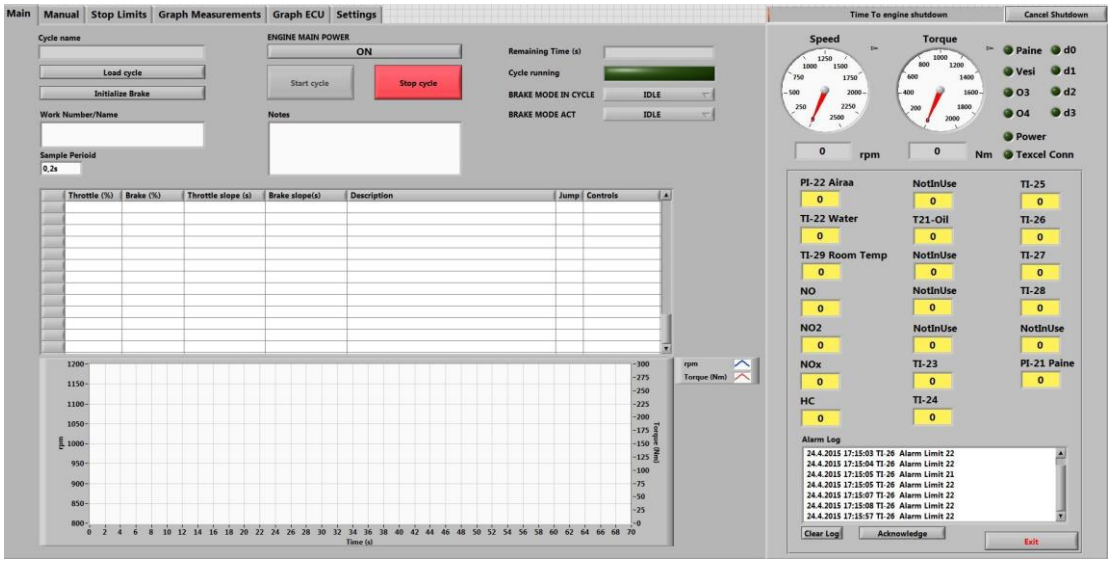


KUVA 11. Cycle Managerin päänäkymä (arvot eivät vastaa oikeaa tilannetta)

Tässä ohjelmassa syklille voidaan antaa nimi ja määritellä ajomoodi, minkä jälkeen sykli voidaan ohjelmoida antamalla taulukon kenttiin kesto kierrosohjeelle ja vääntömomentille. Lisäksi ohjelmoitua sykliä voidaan tarkastella ikkunan alalaitaan piirtyvästä kuvaajasta. Edellä mainittujen lisäksi ohjearvoille voidaan määritellä nousu- ja laskuajat slope-kentässä. Jump-toiminnolla voidaan suorittaa hyppyjä haluttuun kohtaan, jotta sykliä voidaan ajaa monta kertaa peräkkäin ohjelmoimatta samaa pätkää uudestaan. Control-kentässä voidaan hallita neljää ohjelmoitavaa digitaalilähtöä, joiden avulla katalysaattorille voidaan luoda erilaisia testiolosuhteita esimerkiksi ruiskuttamalla vettä. Lopuksi sykli voidaan tallentaa tiedostoon myöhempää käyttöä varten.

Kuvassa 12 on esitetty itse testausohjelman päänäkymä, josta Cycle Managerilla tallennettuja syklejä voidaan ajaa. Sykli näkyy taulukossa hieman eri tavalla kuin Cycle Manager -ohjelmassa, sillä sykli on purettu yhden sekunnin askeleisiin. Näin ajon aikana ohjearvoja voidaan tarkastella tarkemmin, sillä myös nousu- ja laskukohtien ohjearvot, jotka ohjelma laskee itse, ovat näkyvissä. Kun käyttäjä lataa syklin, syötetyt arvot tarkistetaan moottorin parametreja vasten. Jos sykliin on vahingossa lipsahtanut esimerkiksi liian pieni nopeus tai liian

suuri vääntö, sykli jätetään lataamatta ja käyttäjälle ilmoitetaan, millä rivillä virhe sijaitsee. Esimerkiksi kuvan 11 sykli ei menisi läpi, koska viimeisen rivin kierrosohje on hyvin alhainen.



KUVA 12. Testausohjelman Main-näkymä

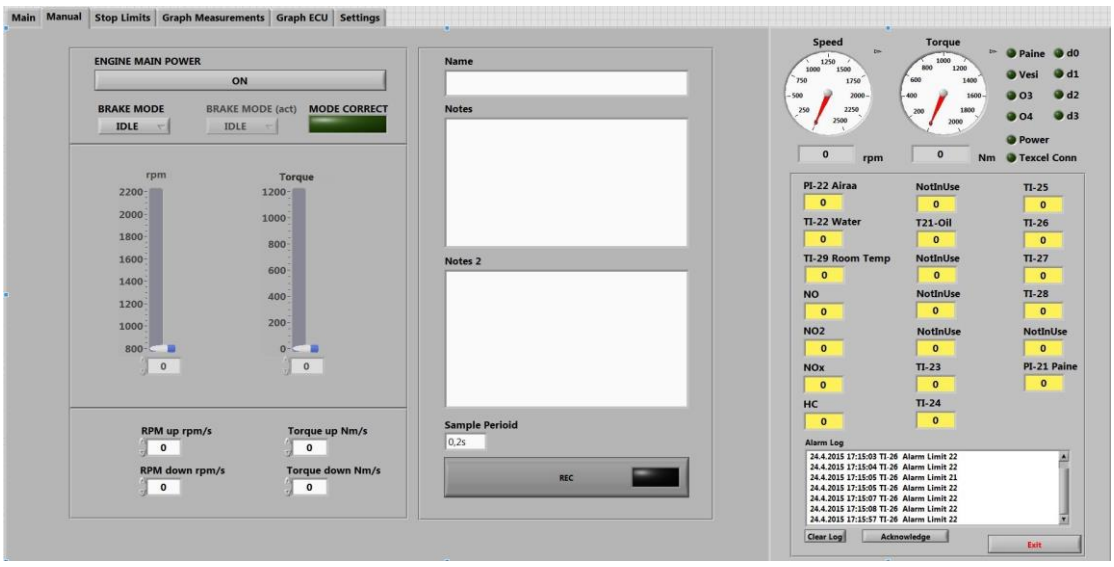
Ennen syklin käynnistystä testiajolle voidaan antaa nimi ja kuvaus, minkä lisäksi voidaan valita lokin näytteenottoväli. Initialize Brake -napilla moottori voidaan laittaa tyhjäkäynnille, mikäli se on esimerkiksi manuaaliajon seurauksena jäänyt pyörimään kovemmalla nopeudella. Samalla jarru asetetaan syklissä käytettyyn moodiin. Ohjelmassa päädyttiin käyttämään vain yhtä ajomoodia, sillä käytössä ollut ECU ei asiakkaan mukaan vastaanottanut käskyjä analogiatulon kautta. Tästä johtuen ainoa tuettu ajomoodi on kierrokset ja vääntömomentti eli moottorille annetaan kierrosohje ja jarrulle vääntömomentti. Ohjelmassa on kuitenkin tuki muillekin moodeille, mikäli testipaikan laitteisto muuttuu ja ne halutaan käyttöön. Syklin etenemistä voidaan seurata taulukosta ja kuvaajasta, jonka päällä näkyy kursori osoittamassa kohtaa, jossa ollaan menossa.

Main-ikkunan oikealla puolella oleva kenttä on näkyvissä koko ajan valitusta välilehdestä riippumatta. Kenttä sisältää tärkeimmät mittaustiedot ja tulojen sekä lähtöjen tilanosoituksen. Tämän lisäksi alalaidassa on yksinkertainen hälytysloki, johon kirjoitetaan viestejä mikäli joku käytössä olevista hälytys- tai pysäytysrajoista ylitetään.

Syklin loppumisen jälkeen moottorin annetaan käydä tyhjäkäynnillä viiden minuutin ajan, minkä jälkeen moottori sammutetaan. Tästä ilmoitetaan käyttäjälle oikeaan yläkulmaan ilmestyvällä edistymispalkilla, jonka Cancel-painikkeella sammutus voidaan tarvittaessa pysäyttää.

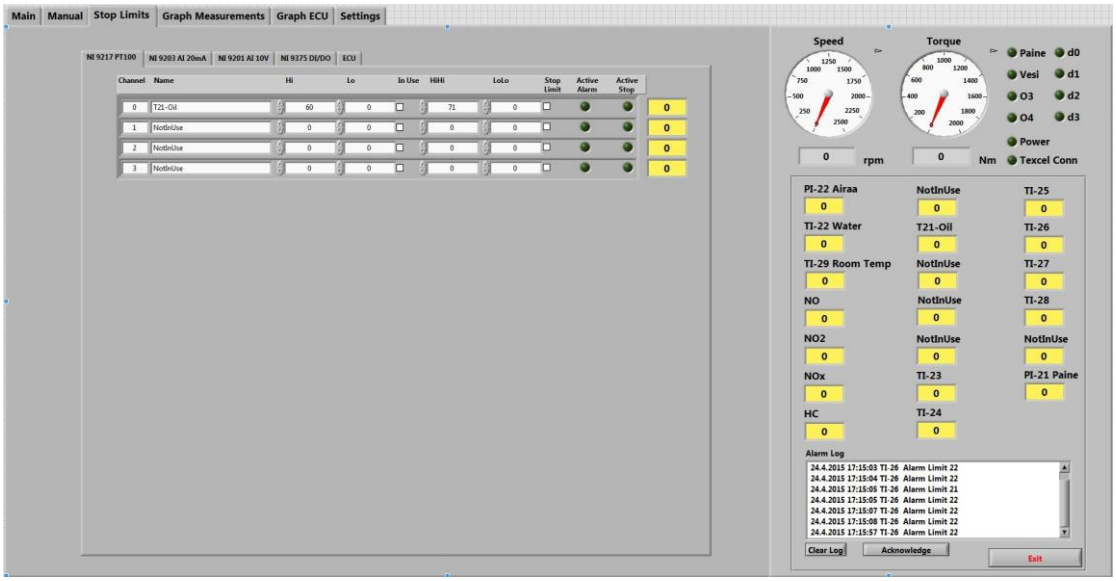
Manual-välilehdellä käyttäjä pystyy ajamaan testauslaitteistoa käsin. Näkymä on esitetty kuvassa 13. Aluksi käyttäjä valitsee ajomoodin. Kuten aikaisemmin on kerrottu käyttöön tuli vain yksi ajomoodi, minkä lisäksi Idle-tila, joka tarkoittaa moottorin tyhjäkäyntiä ja 0 Nm:n vääntöpyyntöä jarrulle. Kun ajomoodi valitaan, liukupalkkeihin haetaan moodia vastaavat ohjattavien suureiden nimet ja skaalat. Mode correct -merkkivalo syttyy, kun haluttu ajomoodi vastaa todellista. Liukupalkeissa on kaksi kursoria. Valkoista kursoria vetämällä käyttäjä voi valita lopullisen asetusarvon. Alhaalla näkyvissä kentissä voidaan kuitenkin määritellä, kuinka nopeita nousuja tai laskuja halutaan käyttää. Tämä todellinen ohjausarvo kyseisellä hetkellä ilmaistaan sinisellä kursorilla, joka lähtee etenemään kohti valkoista määritetyllä nopeudella.

Manuaaliajolle voidaan tarvittaessa käynnistää mittaustietojen tallennus REC-napilla, jolloin data tallennetaan vastaavalla tavalla kuin normaalissa sykliajossa.



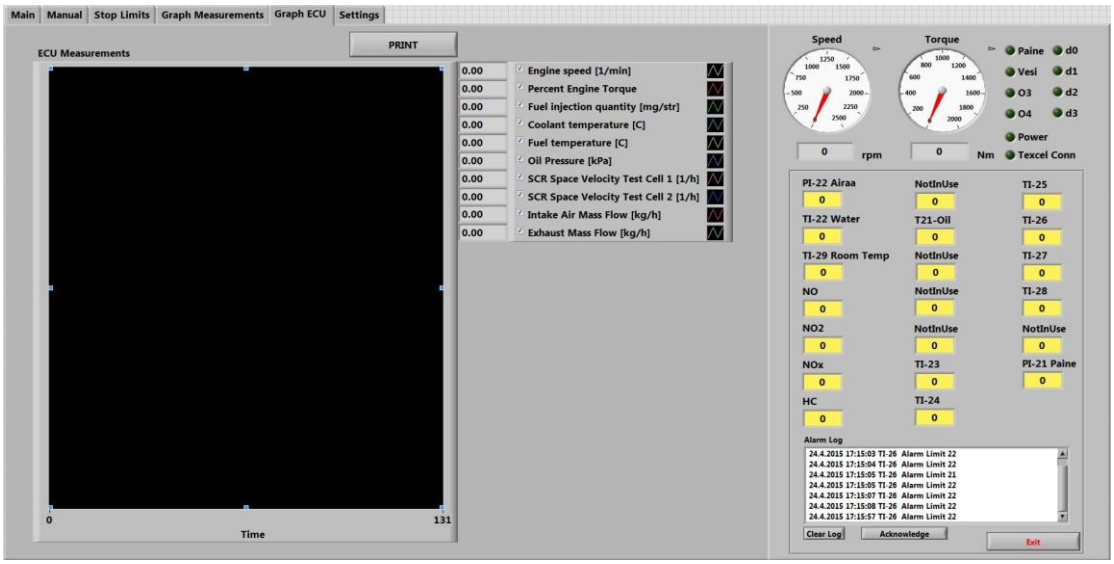
KUVA 13. Manual-välilehti

Seuraavalla eli Stop Limits -välilehdellä käyttäjä pystyy ottamaan käyttöön hälytys- ja pysäytysrajat mittauksille, kuten kuvasta 14 nähdään. Asiakkaan toiveesta kanavien mittaukset pystyy myös nimeämään uudelleen, koska kokoonpano saattaa joskus vaihdella. Nimet päivittyvät tietysti myös lokeihin, kuvaajiin ja mittausten indikaattoreihin. ECU-mittauksissa voidaan ottaa käyttöön vain hälytysrajat ja moottorin sammutus on jätetty ECU:n huoleksi. Toiveena oli myös, että rajoja pystytään muuttamaan syklin ollessa ajossa, koska joskus rajoja joudutaan hakemaan kokeellisesti. Tätä helpottamaan on lisätty mittausarvon osoitus myös kanavan rivin loppuun.



KUVA 14. Stop Limits -välilehti

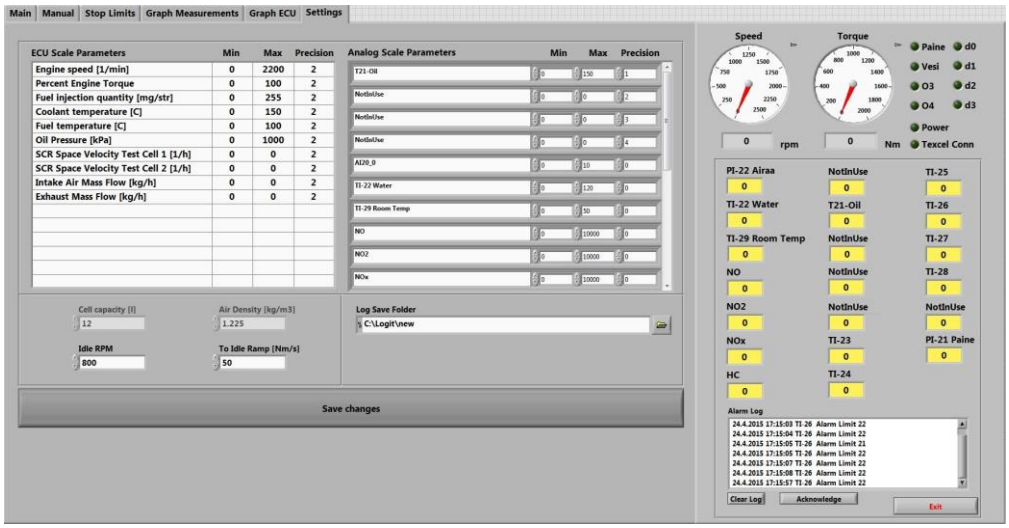
Graph Measurements- ja Graph ECU -välilehdiltä, jotka on esitetty kuvassa 15, voidaan tarkastella mittausten kuvaajia. Tarkastelun selkeyttämiseksi tarpeettomat mittaukset voidaan piilottaa. Mittausarvo näytetään vielä nimen edessä seuraamisen helpottamiseksi. Kuvaajan voi tarvittaessa tulostaa.



KUVA 15. Graph ECU -välilehti

Settings-välilehdellä, joka on näkyvissä kuvassa 16, voidaan asettaa muutamia käytön kannalta tarpeellisia asetuksia. Taulukoista mittauksille voidaan asettaa asteikot kuvaajia varten, minkä lisäksi tarkkuus eli desimaalien määrä raporttitiedostoon. Raporttitiedostoille voidaan valita myös tallennuskansio. Lisäksi voidaan määritellä Idle-tilan tyhjäkäyntinopeus ja momentin pudotus sekuntia kohti tilanteessa, jossa siirrytään ajosta Idle-tilaan. Asia on hoidettu näin, jotta vältettäisiin liian äkillinen kuorman pudotus. Cell capacity ja air density ovat parametreja SCR space velocity -laskentaa varten. Tämä laskenta jätettiin käyttöönotossa kuitenkin vielä pois, puuttuvien antureiden takia.

Settings-, Manual- ja Stop Limits -välilehtien asetukset tallennetaan tietysti tiedostoon, eikä asetuksia tarvitse syöttää uudestaan jokaisella käynnistyskerralla.



KUVA 16. Settings-välilehti

6 KÄYTTÖÖNOTTO

Käyttöönotto aloitettiin testaamalla yhteys jarruun. Jarrusta löytyy simulointitila, jonka avulla toimintaa voidaan testata. Simulointitilassa jarru ottaa vastaan komentoja ja vastailee asianmukaisesti lähettämättä niitä kuitenkaan oikeasti laitteistolle. Tässä vaiheessa huomattiin pieni erikoisuus, sillä jarrun viestit olivatkin hiukan erilaisia kuin ohjekirja antoi ymmärtää. Muutos oli sellainen, että jarru lähetti mittaustietojen lisäksi ylimääräisen kentän, joka sisälsi juoksevan numeron. Tästä syystä sarjaliikenneajuriin jouduttiin tekemään pieniä muutoksia, minkä jälkeen tiedot saatiin luettua oikein, ja sijoitettua oikeisiin muuttujiin. Samana päivänä tehtiin myös pieniä lisäyksiä ja korjauksia käyttöliittymään.

Seuraavana päivänä siirryttiin testaamaan CAN-liikennettä, joka lähti toimimaan pääosin hyvin ja moottorin lähettämät viestit saatiin luettua odotetulla tavalla. Ohjauksen kanssa sen sijaan oli aluksi ongelmia ja huomattiin, että ohjausviestin lähetysväli pitää olla hyvin tarkasti ohjekirjan mukainen. Samalla tarkistettiin antureiden kytkentöjä ja tehtiin muutamia muutoksia, sillä anturit erosivat hieman odotetusta. Kun yhteys moottoriin todettiin toimivaksi, oli tarkoitus siirtyä testaamaan moottorin ja jarrun yhteistoimintaa. Tässä vaiheessa kohdattiin kuitenkin ongelmia. Jarru vaati uudelleenkäynnistystä, kun simulointitilasta siirryttiin oikeaan ajotilaan. Uudelleenkäynnistyksen jälkeen jarru ei kuitenkaan enää suostunut kommunikoimaan automaatio-ohjaimen kanssa. Sarjaliikenneyhteyden asetukset tarkistettiin useampaan kertaan ja kokeiltiin eri kaapeleita, mikä ei kuitenkaan johtanut toivottuun lopputulokseen. Tässä vaiheessa jo hiukan pelättiin, että jarrun sarjaporttia ohjaava piiri olisi mennyt rikki esimerkiksi maalenkin ansiosta, ja vastaavaa korttia etsittiin laboratorion varaosista.

Seuraavana päivänä avuksi otettiin oskilloskooppi ja jarrun sarjaportin signaalia alettiin mitata. Vähän yllättävästi signaali näytti kuitenkin normaalilta ja jarru kytkettiin tietokoneeseen, minkä jälkeen porttia luettiin terminaaliohjelmalla. Liikenne olikin alkanut taas toimia ja viestit tulivat selkokielisinä perille. Tässä vaiheessa pääteltiin, että sammutuksen jälkeen jarrun täytyy antaa olla sammutettuna pidemmän aikaa tai sarjaliikenne lakkaa toimimasta.

Kun yhteydet saatiin toimimaan, siirryttiin ajamaan sykliä ja kokeiltiin yhtä lyhyttä testiohjelmaa. Tässä vaiheessa huomattiin, että sarjaliikenneajuri vaatii vielä lisää optimointia, jotta reagointi olisi tarpeeksi nopeaa. Tämän jälkeen automaatio-ohjaimeen kytkettiin loputkin anturit, digitaalilähdöt ja ohjelmaan tehtiin skaalaukset, jotka muuttavat analogiaviestit mitattaviksi suureiksi. Tämän jälkeen laitteistoa testattiin lisää ja samalla ohjelmaan tehtiin muutamia lisäyksiä ja korjauksia. Lopuksi kaikki ohjelmat käännettiin ajettaviksi ilman kehitysympäristöä. Tämän jälkeen RT-ohjelma ladattiin automaatio-ohjaimeen ja asetettiin käynnistymään, kun laite laitetaan päälle. Tässä vaiheessa kuitenkin tuli yllättäviä ongelmia. Yhteys CAN- ja DI/DO-moduuleihin katosi, mikä johtui siitä, että muutamia kilpailutilanteita ei huomioitu tarpeeksi. Osa ongelmista saatiin ratkaistua ja osa jäi viikonlopun jälkeen ratkaistavaksi Protaconin Jyväskylän toimistolle. Lopulta ongelmat korjattiin ja testauslaitteisto saatiin onnistuneesti käyttöön.

7 YHTEENVETO

Työn tarkoituksena oli toteuttaa asiakkaan vaatimukset täyttävä testilaitteisto, jonka ytimenä toimi CompactRIO-sarjan automaatio-ohjain. Projekti päättyi lopulta onnistuneesti, lopun yllättävistä ongelmista huolimatta, ja testilaitteistoa pystyttiin ajamaan asiakkaan toivomilla nopeuksilla.

Ohjelmaa pystyisi parantamaan lisäämällä siihen ajomoodeja, kuten kaasun kulma ja momentti sekä kaasun kulma ja rpm, mutta kyseisellä laitteistolla se ei välttämättä ole mahdollista, koska ECU ei jostain syystä hyväksynyt analogista viestiä, jonka avulla kaasun asento välitetään. Ohjelman käytettävyyttä voisi parantaa parilla lisäyksellä. Syklien tuonti Excel-tiedostosta voisi olla hyödyllinen, jos halutaan käyttää vanhoja syklejä, jotka ovat tehty eri ohjelmalla. Toinen pieni lisäys voisi toimia niin, että käyttäjä voisi syöttää minuutteina ajan, kuinka kauan sykliä halutaan ajaa. Kyseiset lisäykset tullaan mahdollisesti myös tekemään.

Asioita, joissa olisi parantamisen varaa seuraavia projekteja silmällä pitäen, ovat ainakin ohjelman selkeys ja kilpailutilanteiden ottaminen huomioon. Ohjelman selkeys on erityisen tärkeää ylläpidettävyyden ja vianhaun kannalta, joten isommat kokonaisuudet olisi kannattanut suunnitella paremmin etukäteen. Projektin aikana tuli opittua paljon varsinkin LabVIEW-ympäristöstä, ja jatkossa asiat pystyisi tekemään suunnitelmallisemmin, koska erilaiset toimintatavat tulivat tutuksi.

LÄHTEET

1. Dinex ostaa Ecocatin. Ecocat. 2013. Saatavissa: http://www.eco-cat.com/pdf/Dinex_ostaa_Ecocatin.pdf Hakupäivä: 25.5.2015

2. 3rd Generation Series 6-Cylinder Diesel Engine. 2008. AGCO Sisu Power. Saatavissa: http://www.agcopower.com/@Bin/1635426/SisuDiesel_6cylEng.pdf Hakupäivä 19.5.2015.

3. ECU Designing and Testing using National Instruments Products. 2009. National Instruments. Saatavissa: http://www.ni.com/white-paper/3312/en/ Hakupäivä 27.5.2015.

4. Control Area Network (CAN) Overview. 2014. National Instruments. Saatavissa: http://www.ni.com/white-paper/2732/en/ Hakupäivä 19.05.2015.

5. CAN bus. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/CAN_bus Hakupäivä 19.0.2015.

6. RS-232. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/RS-232 Hakupäivä 19.5.2015.

7. Serial port. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/Serial_port Hakupäivä 19.5.2015.

8. Parity bit. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/Parity_bit Hakupäivä 19.5.2015.

9. LabVIEW. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/LabVIEW Hakupäivä 19.5.2015.

10. Race condition. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/Race_condition Hakupäivä 19.5.2015.

11. FPGA Fundamentals. 2012. National Instruments. Saatavissa: http://www.ni.com/white-paper/6983/en/ Hakupäivä 19.5.2015.

12. Timing Violations When Compiling Hybrid Mode Bitfile with Specific Modules Using Vivado Compilation Tools. 2015. National Instruments. Saatavissa: http://digital.ni.com/public.nsf/allkb/EE21CD363EDE58E486257E0E0045C9A8 Hakupäivä 19.5.2015.